# Robust Learning Protocol for Federated Tumor Segmentation Challenge

Ambrish Rawat[1], Giulio Zizzo[1], Swanand Kadhe[2], Jonathan P. Epperlein[1], and Stefano Braghin[1]

[1] IBM Research Europe, Dublin, Ireland
{ambrish.rawat@ie, giulio.zizzo2, jpepperlein@ie, stefanob@ie}.ibm.com
[2] IBM Research, Almaden, USA
swanand.kadhe@ibm.com

**Abstract.** In this work, we devise robust and efficient learning protocols for orchestrating a Federated Learning (FL) process for the Federated Tumor Segmentation Challenge (FeTS 2022). Enabling FL for FeTS setup is challenging mainly due to data heterogeneity among collaborators and communication cost of training. To tackle these challenges, we propose **Ro**bust **Le**arning **Pro**tocol (RoLePRO) which is a combination of server-side adaptive optimisation (e.g., server-side Adam) and judicious parameter (weights) aggregation schemes (e.g., adaptive weighted aggregation). RoLePRO takes a two-phase approach, where the first phase consists of vanilla Federated Averaging, while the second phase consists of a judicious aggregation scheme that uses a sophisticated reweighting, all in the presence of an adaptive optimisation algorithm at the server. We draw insights from extensive experimentation to tune learning rates for the two phases.

**Keywords:** Federated Learning · Adaptive Optimisation · Brain Tumor Segmentation.

## 1 Introduction

In this work we investigate a federated learning system for Brain Tumor Segmentation as presented in the Federated Tumor Segmentation Challenge (FeTS) [1,2,3]. Brain Tumor Segmentation is a medical imaging task where given a an MRI scan a model is tasked to produce the segmentation demarcating the regions corresponding to a tumor namely, enhancing tumor (ET), tumor core (TC) and whole tumor (WT). Traditionally, a deep learning model with a U-Net architecture [4] is trained for this task. The data for training such models often resides with different institutions which prohibits the use of classical Machine Learning (ML) pipelines that require the data to be available centrally at one location. Therefore, in such situations one may resort to the privacy-preserving paradigm of FL [5] for training the model, which is the focus of FeTS. This challenge presents two tasks - the first involves algorithms for FL orchestration for improved model convergence, and the second is testing the generalisability of a model in the "wild" on the data of clients who did not participate in the original federation. We primarily focus on the first task and investigate a combination of approaches for client- and server-side optimisation schedules, parameter aggregation and fine tuning to help with model convergence.

## 2 FeTS Setup

The federation consists of 23 different institutions [1,2,3] seeking to collaboratively learn a U-Net model [4] for brain tumor segmentation. Each participating institution owns a variably sized data partition which resides privately with the host. The details of the partitioning are specified in `partitioning1.csv` where one can notice a skewed distribution with two collaborators, namely and together hold a major chunk of the total data. On top of this, the classical iid assumption of machine leaning often goes for a toss and the respective partitions may have highly non-iid characteristics. Such skewed distributions are not foreign for federated setups and often the learning protocols are appropriately modified to account for the heterogeneity. In order to moderate the heterogeneity, the challenge also presents an additional partitioning where some institutions are further split based on the tumour size, resulting in a more balanced distribution across 33 clients as specified in `partitioning2.csv`.

More generally, such cross-silo FL setups consists of $M$ clients, where $m^{\text{th}}$ client owns data $D_m = \{(x_i, y_i)\}_{i=1}^{n_m}$ with $n_m$ samples. The training is performed iteratively across multiple FL rounds. In round $t$, first the central server or aggregator selects a set $C_{(t)}$ of clients, referred to as a collaboration, for training. Second it broadcasts the current set of parameter vector W $=\{w_{t,j}\}$ to all $N$ clients for computing a set of validation metrics (for brevity, we will often drop the dependence on $j$ and refer to a single parameter as $w$). Third the clients in the collaboration $C_{(t)}$ train the model on their local data partitions to obtain the updated parameter $w_{t+1}^c$ along with a set of post-training validation metrics, both of which they share with the aggregator. Finally the aggregator combines the weight vectors from different clients typically with weighted averaging (FedAvg) as, $w_{t+1} = \sum_{c \in C_{(t)}} n_c w_{t+1}^c / N_{C_{(t)}}$ where $N_{C_{(t)}} = \sum_{c \in C_{(t)}} n_c$ is the sum of samples across clients in collaboration $C_{(t)}$. The federation is performed for a total of $T$ rounds with suitable measures like early stopping to help reduce the generalisation error.

***Metrics.*** There are two metrics tracked as part of FeTS setup - Dice similarity score (DSC) which measures the overlap between predicted and ground truth segmentations and Hausdroff distance (HD) which accounts for the distance between segmentation boundaries. The details for these metrics are described in [1]. During the FL training, FeTS maintains the checkpoint of model which achieves the best DSC on the validation set which itself is set locally by each client with a classical 80-20 split.

***Modelling Challenges.*** There are two key challenges in enabling federated learning for such settings. The first stems from the data distributions across different clients which are typically not iid. This often leads to diverging updates during training rounds which hamper model convergence. And second is the communication cost of training rounds and the presence of stragglers in the federation, both of which add an overhead for the learning process.

Typically FL setups either belong to a cross-silo or a cross-device setting. The former comprises of a clientele of the order of 100 participants each holding a relatively large data set while the latter could have as many as a billion clients with only few data samples associated with each participant. A synchronous orchestration with all clients participating in every round adds a massive communication overhead which can adversarially affect the rate of convergence. On the other hand, ignoring certain clients could bias the resultant model towards specific modes thereby deteriorating their generalisation capability.

***Practical Challenges.*** While FeTS presents an interesting setup for exploring a combination of schemes for FL orchestration, it is worth noting that there are inherent assumptions in the system. An understanding of these assumptions helps in scoping out the playing field and shaping strategies for algorithm design and experimentation. For instance, the APIs for Task 1 limit access to local training protocols of participating clients which limits the the applicability of approaches like FedProx [6] and Scaffold [7] to tackle the non-iidness in the system. Similarly, the available deployment of FeTS requires relatively large compute, of the order of 300 GB of CPU RAM optionally with at least 16 GB of GPU memory. Additionally, each training rounds can take up to 7 hours which can delay the feedback often required for rapid prototyping, hyperparameter optimisation and prohibit re-runs for marginalising the experimentation noise. Finally, it should be mentioned that the orchestration maintains a counter for total simulated time which accounts for the different costs in typical FL process including communication cost, training time and cost related to computation of validation metrics. The script exits when the total simulated time reaches one week. It is also worth mentioning that the specified FL plan for FeTS involves global computation of metrics i.e. each client, irrespective of their participation in the collaboration, computes the set of validation metrics on the latest aggregated model. This, as we discuss in Section, can be beneficial for devising both client selection and parameter aggregation strategies. In summary, we work within these constraints and devise schemes that can help improve the rate of model convergence within the permissible simulated time.

***FeTS 2021.*** The challenge was also hosted in 2021 [1] with some notable differences. To our understanding this year's challenge includes data from additional institutions potentially exacerbating the non-iidness in the system. We also note that Cost Weighted Averaging [8] and Adaptive Weight Averaging [9] resulted in winning solutions for the posed challenge which we explore within our overall FL plan as described in Section 3.

## 3 Our Approach

***Summary of our approach.*** We propose to use a combination of server-side adaptive optimisation and judicious parameter aggregation in a two phase process with all clients participating in every round. In the first phase, the server aggregates the weights with vanilla FedAvg and in the second it employs a judicious aggregation scheme which uses a more sophisticated re-weighting (we discuss several judicious aggregation schemes later). In both phases, server-side adaptive optimisation (e.g., server-side Adam described later) is used. The learning rates are appropriately adjusted for the two phases. We summarise our approach in Fig. 1. In the remaining section, we describe our thinking behind the proposed approach and provide details of its various components.

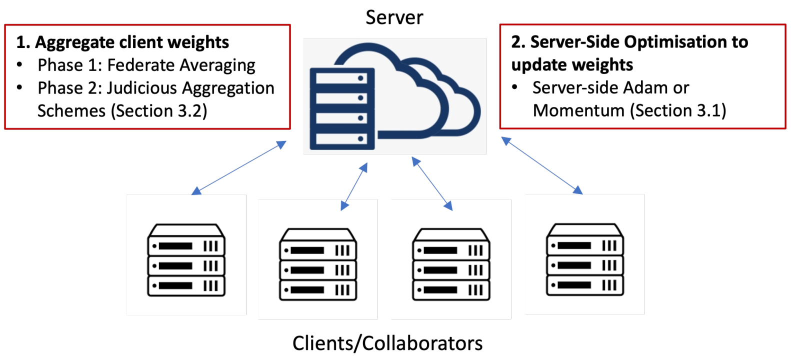
Server
1. Aggregate client weights
• Phase 1: Federate Averaging
• Phase 2: Judicious Aggregation Schemes (Section 3.2)
2. Server-Side Optimisation to update weights
• Server-side Adam or Momentum (Section 3.1)
Clients/Collaborators

**Fig. 1.** Summary of our approach: a combination of server-side adaptive optimisation and judicious parameter aggregation in a two phase process with all clients participating in every round.

FeTS presents a highly constrained and challenging scenario where both modelling and experimentation challenges need to be addressed simultaneously. These constraints limit the use of brute-force approaches for algorithm design, experimentation, and validation as well as hyperparameter optimisation. We therefore break the overall task into smaller components with a seat-of-the-pants strategy. First, we run the vanilla FedAvg algorithm with the default settings and make some observations. This involves every client participating in every round of the collaboration. Based on these observations we conjecture a working theory for this setup and use it to devise experiments. In particular, we note the behaviour of the optimisation and adopt a suitable learning rate schedule to fit this scenario. Additionally, we also adopt the server-side optimisation with adaptive schemes to help speed up convergence. We then look at a set of different aggregation schemes which are inspired from the successful solutions of FeTS 2021. Finally, we consider a two-phase process where we consider aggregation schemes as part of a fine tuning strategy. In this approach we first obtain a reasonable initialisation within first few FL rounds with vanilla FedAvg and follow it up with *fine-tuning* phase for the remaining time. For all of these experiments we only adopt the strategies for weights and biases in the U-Net. All other parameters are aggregated with weights proportional to sample size. It is worth noting that, even though we primarily focus on all clients participating in every round, our approach can in general be adapted to include client selection.

***Default Setup.*** We first run the experiment with the default setting which consist of: all clients participating in every round, vanilla federated averaging, and fixed hyperparameters of learning rate and epochs per round across all rounds. We note the following observations from this experiment - 1) while there are two different partitions available we note that `partitioning2.csv` naturally provides a better fit with data parallelism, 2) the optimisation is not stable with an oscillating objective during the optimisation, 3) we experiment with local epochs of 1 and 3 and do not note any remarkable difference in DICE scores. We also note that it takes an average of 700 minutes (of simulated time) per round with all clients participating. Thus, with all 33 client training for one local epoch in each round, one can perform a federation for up to 16 training rounds within the 1 week limit specified in the setup.

## 3.1 Server-Side Optimisation

In order to dampen the oscillations we explore the use of adaptive optimisation strategies. An FL setup consists of two different leaning rates - a client side learning rate $\lambda_c$ and a server side learning rate $\lambda_s$. In general, at round $t$ the server first computes an aggregate of obtained updates $\hat{w}_{t+1}$ from the clients using the specified aggregation scheme, which is used to form a proxy $\varDelta_t = w_t - \hat{w}_{t+1}$ for the gradient from the server's perspective. It then uses an optimisation strategy to modify its parameter state with the obtained delta. Thus, vanilla FedAvg can be thought of as performing a server side Stochastic Gradient Descent (SGD) $w_{t+1} = w_t - \lambda_s \varDelta_t$ with $\lambda_s = 1.0$. However, adaptive strategies like momentum-based aggregation or even server-side Adam can be used to accelerate the convergence and dampen the oscillations [10,11]. It is worth pointing out that in FeTS the client-side optimisation is fixed as Adam [12]. The works of [10] and [11] provide an in-depth analysis of optimisation schemes for federated setup with some useful insights. We focus on two key takeaways from this work - they recommend jointly tuning $\lambda_c$ and $\lambda_s$ as they observe that the accuracy often follows a staircase pattern for adaptive schemes, and to decay the client-side learning rates when clients take more than a few gradient steps in their local optimisation. Finally, they also note that momentum-based adaption is more sensitive to choice of learning rates, while FedAdam and FedYogi [10] are more stable with respect to these choices.

For this experiment we stick with the classical sample-size-based weighting for aggregation and explore the use of adaptive optimisation strategies at the server end. We later adopt the learning from this experiment across other aggregation schemes. For convenience we refer to this intermediate parameter state as $\hat{w}_t$ which refers to the parameter obtained after using an aggregation scheme at the server end. This is subsequently used in combination with different optimisation approaches

- **OptAlgo A**: server-side SGD with $\lambda_s = 1.0$ combined with step-wise constant learning rates for clients across different rounds where $\lambda_c = 5 \cdot 10^{-5}$ for first 7 rounds followed by $\lambda_c = 5 \cdot 10^{-6}$ for the remaining rounds.
- **OptAlgo B**: server-side momentum where a moving average is maintained with respect to change in $\Delta_t$ which is in turn used to update the parameter state. The update can be summarised as,

$$\begin{aligned} \Delta_t &= w_t - \hat{w}_{t+1} \\ m_t &= \beta m_{t-1} + \Delta_t \\ w_{t+1} &= w_t - \lambda_s m_t \end{aligned} \tag{1}$$

  This has been explored in [13] and an implementation is available in FeTS. We use a $\beta$ of 0.9 for this approach and fix $\lambda_s$ as 0.1 and $\lambda_c = 5 \cdot 10^{-4}$ for first 7 rounds followed by $\lambda_c = 5 \cdot 10^{-5}$ for the remaining rounds.
- **OptAlgo C**: server-side Adam adopts a parameter-specific update by upweighting updates for parameters which receive sparse updates

$$\begin{aligned} \Delta_t &= w_t - \hat{w}_{t+1} \\ m_t &= \beta_1 m_{t-1} + (1 - \beta_1)\, \Delta_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2)\, \Delta_t^2 \\ w_{t+1} &= w_t - \lambda_s \frac{m_t}{\sqrt{v_t} + \tau} \end{aligned} \tag{2}$$

  FedAdam has previously been investigated in [10] where the authors fix $\tau$ as 0.001 and $\beta_1$ and $\beta_2$ as 0.9 and 0.99 respectively. They further experiment with a range of values for $\lambda_c$ and $\lambda_s$ and note that it often follows a staircase pattern. Following some initial experiments we note that $\lambda_s$ of 0.001 and $\lambda_c$ of $5 \cdot 10^{-4}$ provides stable updates. While this scheme differs slightly from the original Adam implementation where the $\tau$ is added as part of the square root and $m_t$ and $v_t$ are scaled with a decay for every new update, we didn't find an empirical impact on the algorithm when ran for few rounds.

It was clear in our experimentation and as is shown in Figure 2 that adaptive schemes of OptAlgo B and OptAlgo C help dampen the oscillations observed for OptAlgo A. We would like to emphasise that we experimented in an adhoc fashion with manual intervention at different stages. For example, in some experiment runs that lower the $\lambda_c$ further by a factor of 10 also helped with convergence. We suspect that normalising the nuber of updates across clients or reducing the local epochs across rounds with a decay factor can further help with the convergence. In general, our observations were consistent with those made in [10] and [11] - both OptAlgo B and OptAlgo C improved convergnence, and

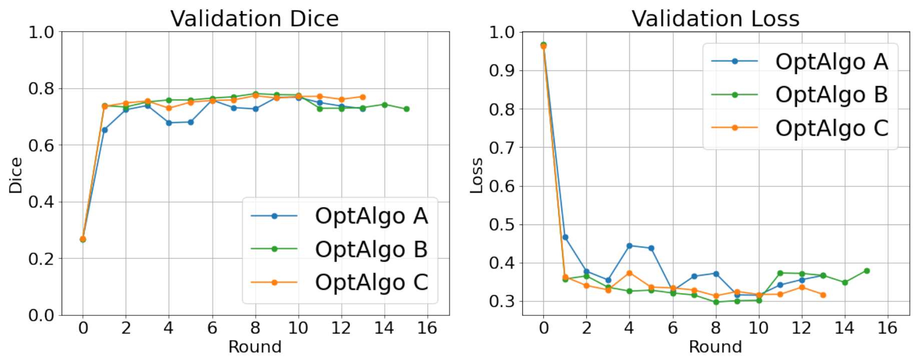

**Fig. 2.** Performance of Optimisation Algorithms with FedAvg for weight aggregation.

OptAlgo C was the most stable with respect to choice of $\lambda_c$ resulting in similar convegrence behaviour across a range of different values for $\lambda_c$ and $\lambda_s$. However, the absence of rigorous experimentation with hyperparameters can potentially result in an apples-vs-oranges comparison. We therefore take the teachings with a pinch of salt and only draw some high level conclusions as in the absence of rigorous testing they only serve as guiding approaches for our subsequent experiments.

## 3.2 Parameter Aggregation Methods

Inspired from the approaches in FeTS 2021, we also experiment with a few different parameter aggregation schemes. These refer to the set of algorithms that are used to obtain $\hat{w}_{t+1}$ from a set of client updates $\{w_t^c\}_{c \in C_{(t)}}$ received in round $t$. We replicate the setup of [8] and [9] to the best of our understanding and experiment with some modifications which are described below.

- **CostWAgg** : Cost Weighted Aggregation (CostWAgg) [8] uses the change in the validation loss across different rounds to guide the weighting during parameter aggregation. More specifically, for each client $c \in C_{(t)}$ it first computes the normalised ratio $r_c = l_c(w_{t-1}^c)/l_c(w_t^c)$ of the local validation loss (after training) from the previous round $l_c(w_{t-1}^c)$ with the one obtained for the current round $l_c(w_t^c)$, thereby giving larger weight to updates which resulted in larger decrease in loss. They further take a convex combination of the cost-based weight and sample-size-based weight to obtain the final parameter vector. In their scheme all clients participate in every round, i.e. $C_{(t)} = \{1, \ldots, M\}$ $\forall t$, and train for 10 epochs locally before sharing the updates. The update in round $t$ is obtained as

$$\hat{w}_{t+1} = \sum_{c \in C_{(t)}} \left( \alpha \frac{n_c}{N_{C_{(t)}}} + (1 - \alpha) \frac{r_c}{R_{C_{(t)}}} \right) w_t^c, \tag{3}$$

where $R_{C_{(t)}} = \sum_{c \in C_{(t)}} r_c$ and $\alpha$ is a hyperparameter for balancing the two terms. The authors recommend $\alpha = 0.5$ while remarking that the optimisation wasn't particularly sensitive to the choice of $\alpha$. Also, note that this scheme results in identical weighting across all parameters. We use a variant of this scheme where each client computes the ratio with respect to the local validation loss computed before and after training in the *current* round, and refer to it as **RoundCWAgg**. This ratio can be computed as $l_c(w_{t-1})/l_c(w_t^c)$. Note that while CostWAgg requires the use of $l_c(w_{t-1}^c)$ which is only available for clients which participated in the previous training round, RoundCWAgg doesn't require the previous loss for the current collaboration, since $l_c(w_{t-1})$ is available for all clients in the federation. We use $\alpha = 0.1$ to upweight the loss based contribution for RoundCWAgg.

- **SimAgg:** Adaptive Weighted Aggregation Policy [9] proposes a parameter weighting scheme that upweights contributions that are close to the average update. First, for each client $c$ it computes an inverse of the absolute difference between client parameter $w_t^c$ and the mean of the collaboration. These are normalised across clients to obtain the client-specific weighting factor $u_c^w$ for the parameter $w$ where clients whose updates are close to the average are assigned larger weights. This is then used in combination with sample-size-based weights $\nu_c = n_c/N_{C_{(t)}}$ to form the final weights with either an additive operator $u_c^w + \nu_c$ or a multiplicative operator $u_c^w \cdot \nu_c$. To our understating, the former is referred to as Regularised Aggregation Policy (RegAgg) and the latter as Similarity Weighted Aggregation (SimAgg) Policy. The authors use random sampling for selecting clients for collaboration (while ensuring fair selection of all clients) and remark that SimAgg with 5 local epochs provided them the best results during their experimentation. SimAgg updates every scalar weight in the weight vector $\hat{w}_{t+1}$ individually. We use $\omega = \hat{w}_{t+1}$ to reduce clutter.

$$\begin{aligned} \omega_{\text{mean}} &= \frac{1}{\#C_{(t)}} \sum_{c \in C_{(t)}} w_t^c \\ u_c^w &= \frac{1/\left|w_t^c - \omega_{\text{mean}}\right|_\tau}{\sum_{i \in C_{(t)}} 1/\left|w_{t,j}^i - \omega_{\text{mean}}\right|_\tau} \\ \omega &= \frac{\sum_{c \in C_{(t)}} \left(u_c^w \cdot \nu_c \cdot w_t^c\right)}{\sum_{c \in C_{(t)}} \left(u_c^w \cdot \nu_c\right)}, \end{aligned} \tag{4}$$

where $|x|_\epsilon := |x| + \epsilon$ avoids divisions by zero; we use $\epsilon = 1 \cdot 10^{-5}$ as specified in [9]. This aggregation scheme results in different weights for different parameters, since $u_c^w$ for every parameter $w$. We also experiment with a modification to this algorithm **SimMedAgg** which uses median ($\omega_{\text{median}}$) for computing $u_c^w$ as opposed to the arithmetic mean.

- **SimCostAgg**: This approach combines the approaches of CostWAgg and SimAgg where the loss ratio $r_c$ is combined with the sample-size weight in a multiplicative way. This results in,

$$\hat{w}_{t+1} = \frac{\sum_{c \in C_{(t)}} (r_c \cdot \nu_c \cdot w_t^c)}{\sum_{c \in C_{(t)}} (r_c \cdot \nu_c)} \tag{5}$$

We run 13 rounds of FL training and use OptAlgoB while comparing these different parameter aggregation scheme. We specifically monitor four metrics - best validation Dice across all labels till round 13, best validation loss till round 13 as well as average validation DICE and validation loss for last five rounds. The results are summarised in Table 1. While in terms of Best Dice (or loss) values many methods are comparable, the Avg Dice (or loss) tells a more relevant story as it sheds light on the stability of the overall optimisation protocol. Mindful of the fact that these numbers weren't obtained over multiple runs, we tentatively conclude that the parameter aggregation schemes provide marginal benefits over vanilla FedAvg (with generally smaller avg loss and higher avg Dice).

**Table 1.** Performance of different parameter aggregation schemes with OptAlgoB for optimmisation. Average values show the stability of the FL process near convergence.

| | Best Dice | Avg Dice | Best loss | Avg loss |
|---|---|---|---|---|
| FedAvg | 0.7771 | 0.7417 | 0.3008 | 0.3522 |
| CostWAgg | 0.7738 | 0.7578 | 0.2984 | 0.3157 |
| RoundCWAgg | 0.7718 | 0.7022 | 0.3074 | 0.4095 |
| SimAgg | 0.7783 | 0.7310 | 0.2920 | 0.3608 |
| SimMedAgg | 0.7569 | 0.7448 | 0.3254 | 0.3478 |
| SimCostAgg | 0.7848 | 0.7595 | 0.2900 | 0.3272 |

### 3.3 Fine Tuning Approaches

In Section 3.1 we noted that OptAlgoB and OptAlgoC help dampen the oscillations and stabilise the server-side optimisation, and in Section 3.2 we noted some benefits in the use of different parameter aggregation schemes over FedAvg. With these two lessons and the proverbial knowledge among practitioners that the benefits of domain specific adaption are more suitable as fine tuning, we conjecture an FL plan - **Ro**bust **Le**arning **Pro**tocol (RoLePRO) that consists of two stages: First, we run vanilla FedAvg for a few initial rounds with all clients participating in each round with either OptAlgoB or OptAlgoC and apprproatiate choices of learning rates $\lambda_s$ and $\lambda_c$. Then, we fine-tune the obtained model with the one of the following aggregation schemes - CostWAgg, SimAgg, TrimmedMean and TopKSimCost, and preferably a reduced set of learning rates. The use of TrimmedMean and TopKSimCost is motivated by an intent to reduce the effect of variance within client updates during aggregation which we explain below.

As described in Section 2 the data split across the different participating institutions is highly imbalanced. Naturally, such scenarios lead to large variance among the client updates in each round. It is worth remarking that within FeTS the institutions share updates after a fixed number of epochs which is common for all clients. Thus, clients perform a varying number of local gradient updates which can exacerbate client drift and result in disparate client updates. The success of CostWAgg and SimAgg in FeTS also bears a mention here. CostWAgg effectively prefers updates with more stable loss changes across different rounds and combines it with sample weights thereby giving a high weight to smoothed updates from large-data clients. Similarly, SimAgg enables preferential weighting by exploiting the geometry of updates in combination with the sample weighting. Effectively, updates which are close the mean-update and correspond to clients with large data receive the largest weights during aggregation. Such weighting for FeTS has also been explored in [14] where they categorised the participating institutions as internal and external depending on their sample contribution and developed different weighting schemes for the two groups. TrimmedMean provides an alternative way to alleviate the variance by only accounting for the geometry where first the set of updates are trimmed by discounting the ones that are too far from the median, and the remaining fraction of filtered updates are averaged (without any weighting). This has been explored in the context of byzantine behaviour in Federated Learning [15,16]. While this approach can be beneficial for softening the aggregation, it often slows down convergence. Also worth noting that for TrimmedMean, the filtering of updates across different clients before the aggregation step is different for different parameters. In TopKSimCost we account for loss values and sample size for this filtering step by first sorting the clients as per their loss-ratio and sample contribution with the score, $n_c/N_{C(t)} \cdot l_c(w^c_{t-1})/l_c(w^c_t)$, similar to SimCostAgg, and then averaging (without weighting) the $k$ best updates with the largest scores. We use a filtering factor of 20% for both TrimmedMean and TopKSimCost. Note that TopKSimCost can be thought of an aggressive or harder version of its soft counterparts SimCostAgg and CostWAgg where instead of down-weighting fruitless contributors one simply discards them and is democratic in its consideration of filtered contributors. Contrary to TrimmedMean, the filtering in TopKSimCost is common across all parameters during a training round.

All of the aforementioned aggregation schemes can result in biased models. For instance, TopKSimCost can lead to a preferential treatment of a select few clients, especially with a high filtering factor. The hyperparameter $\alpha$ controls a similar trade-off for CostWAgg. Such loss-based aggregation has also been explored in other contexts for FL. One example is the Federated Adversarial Training [17] protocol which suffers from highly disparate updates during the training rounds and modifying the weighting scheme helped improve convergence [18]. Similarly, best-$k$ sparsification has been used to combat model poisoning in FL [19].

In Table 2 and 3 we present the results for RoLePro with OptAlgoB and OptAlgoC across different aggregation schemes. We observe that both SimAgg

**Table 2.** Performance of different aggregation schemes with OptAlgoB in the fine-tuning phase of the two-phase learning protocol of RoLePro

| | Best Dice | Avg. Dice | Best loss | Avg. loss |
|---|---|---|---|---|
| SimAgg | 0.8067 | 0.7921 | 0.2639 | 0.2821 |
| CostWAgg | 0.7788 | 0.7681 | 0.3107 | 0.3207 |
| TrimmedMean | 0.7928 | 0.7834 | 0.2764 | 0.2958 |
| TopKSimCost | 0.7965 | 0.7806 | 0.2718 | 0.2957 |

**Table 3.** Performance of different aggregation schemes with OptAlgoC in the fine-tuning phase of the two-phase learning protocol of RoLePro

| | Best Dice | Avg. Dice | Best loss | Avg. loss |
|---|---|---|---|---|
| SimAgg | 0.7958 | 0.7851 | 0.2709 | 0.2905 |
| CostWAgg | 0.7888 | 0.7712 | 0.2880 | 0.3060 |
| TrimmedMean | 0.7588 | 0.7517 | 0.3218 | 0.3314 |
| TopKSimCost | 0.7963 | 0.7700 | 0.2799 | 0.3086 |

and TopKSimCost lead to high performaning models across both OptAlgoB and OptAlgoC. It should be mentioned that apart from aggregation schemes, the optimisation protocols of OptAlgoB and OptAlgoC also enable parameter specific learning rates, therefore it can be challenging to pin-point the source of the observed gains within the complex orchestration of an FL plan. We use this for our final submission.

## 4 Conclusions

It is well known that an FL system has many moving parts from the numerous components within its system design to the set of hyperparameters in the learning algorithm. These can often result in intractable experimentation strategies. In this work we focused on the cross-silo or enterprise setup of FeTS and developed an FL plan to orchestrate the learning over the data residing with 23 participating institutions. We achieved this by studying two different aspects of FL namely, server-side optimisation and judicious parameter aggregation schemes, which we then used to develop a robust learning protocol for the FeTS setup. This protocol prescribes a two phase process where vanilla FedAvg is followed by a fine-tuning phase that is enabled with sophisticated parameter aggregation schemes, all in the presence of an adaptive optimisation algorithm at the server end (such as server-side Adam). While we found the choice of learning rates for the two phases to be crucial for our FL plan, we acknowledge that these hyperparameter choices require thorough analysis. Finally, we hope that this empirical investigation can serve as a guiding document for tackling the underlying challenge of FeTS.

## References

1. Sarthak Pati, Ujjwal Baid, Maximilian Zenk, Brandon Edwards, Micah J. Sheller, G. Anthony Reina, Patrick Foley, Alexey Gruzdev, Jason Martin, Shadi Albarqouni, Yong Chen, Russell Taki Shinohara, Annika Reinke, David Zimmerer, John B. Freymann, Justin S. Kirby, Christos Davatzikos, Rivka R. Colen, Aikaterini Kotrotsou, Daniel S. Marcus, Mikhail Milchenko, Arash Nazeri, Hassan M. Fathallah-Shaykh, Roland Wiest, András Jakab, Marc-André Weber, Abhishek Mahajan, Lena Maier-Hein, Jens Kleesiek, Bjoern H. Menze, Klaus H. Maier-Hein, and Spyridon Bakas. The federated tumor segmentation (fets) challenge. *CoRR*, abs/2105.05874, 2021.
2. G. Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, Jason Martin, Brandon Edwards, Micah J. Sheller, Sarthak Pati, Prakash Narayana Moorthy, Hans Shih-Han Wang, Prashant Shah, and Spyridon Bakas. Openfl: An open-source framework for federated learning. *CoRR*, abs/2105.06413, 2021.
3. Ujjwal Baid, Satyam Ghodasara, Michel Bilello, Suyash Mohan, Evan Calabrese, Errol Colak, Keyvan Farahani, Jayashree Kalpathy-Cramer, Felipe C. Kitamura, Sarthak Pati, Luciano M. Prevedello, Jeffrey D. Rudie, Chiharu Sako, Russell T. Shinohara, Timothy Bergquist, Rong Chai, James A. Eddy, Julia Elliott, Walter Reade, Thomas Schaffter, Thomas Yu, Jiaxin Zheng, BraTS Annotators, Christos Davatzikos, John Mongan, Christopher Hess, Soonmee Cha, Javier E. Villanueva-Meyer, John B. Freymann, Justin S. Kirby, Benedikt Wiestler, Priscila Crivellaro, Rivka R. Colen, Aikaterini Kotrotsou, Daniel S. Marcus, Mikhail Milchenko, Arash Nazeri, Hassan M. Fathallah-Shaykh, Roland Wiest, András Jakab, Marc-André Weber, Abhishek Mahajan, Bjoern H. Menze, Adam E. Flanders, and Spyridon Bakas. The RSNA-ASNR-MICCAI brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. *CoRR*, abs/2107.02314, 2021.
4. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.
5. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.
6. Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze, editors, *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.
7. Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Confer-*

*ence on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.

8. Leon Mächler, Ivan Ezhov, Florian Kofler, Suprosanna Shit, Johannes C. Paetzold, Timo Loehr, Benedikt Wiestler, and Bjoern H. Menze. Fedcostwavg: A new averaging for better federated learning. *CoRR*, abs/2111.08649, 2021.
9. Muhamman Irfan Khan, Mojitaba Jafaritadi, Esa Alhoniemi, Elina Konito, and Suleiman A. Khan. Adaptive weight aggregation in federated learning for glioblastoma segmentation. `https://privasa.eu/tuas-wins-second-position-in-international-machine-learning-challenge/`, 2021.
10. Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
11. Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Agüera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas N. Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horváth, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konečný, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtárik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake E. Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. A field guide to federated optimization. *CoRR*, abs/2107.06917, 2021.
12. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
13. Ece Isik-Polat, Gorkem Polat, Altan Kocyigit1, and Alptekin Temizel1. Federated brain tumor segmentation:multi-institutional privacy-preserving collaborative learning, 2021.
14. Ruichen Luo, Shoubo Hu, and Lequan Yu. Rethinking client reweighting for selfish federated learning, 2022.
15. El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3518–3527. PMLR, 2018.
16. Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8632–8642, 2019.
17. Giulio Zizzo, Ambrish Rawat, Mathieu Sinn, and Beat Buesser. FAT: federated adversarial training. *CoRR*, abs/2012.01791, 2020.
18. Jianing Zhu, Jiangchao Yao, Tongliang Liu, Kunyang Jia, Jingren Zhou, Bo Han, and Hongxia Yang. \$\alpha\$-weighted federated adversarial training, 2022.

19. Ashwinee Panda, Saeed Mahloujifar, Arjun Nitin Bhagoji, Supriyo Chakraborty, and Prateek Mittal. Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 7587–7624. PMLR, 2022.